Securing Systems through Software Reliability Engineering

Taz Daughtrey

Data and Analysis Center for Software

Photo # NH 96566-KN First Computer "Bug", 1945

92

9/9

0800 Anctan started {1.2700 9.037 847 025

1000 " stopped - anctan ✓ 9.037 846 795 correct

13" uc (032) MP - MC ~~1.982147000~~ ~~2.130476415~~ 4.615925059(-2)

(033) PRO 2 2.130476415

correct 2.130676415

Relays 6-2 in 033 failed special speed test

In relay " 11.000 test.

Relays changed

Relay 2145

Relay 3370

1100 Started Cosine Tape (Sine check)

1525 Started Mult + Adder Test.

1545  Relay #70 Panel F (moth) in relay.

First actual case of bug being found.

1630 antangent started.

1700 closed down.

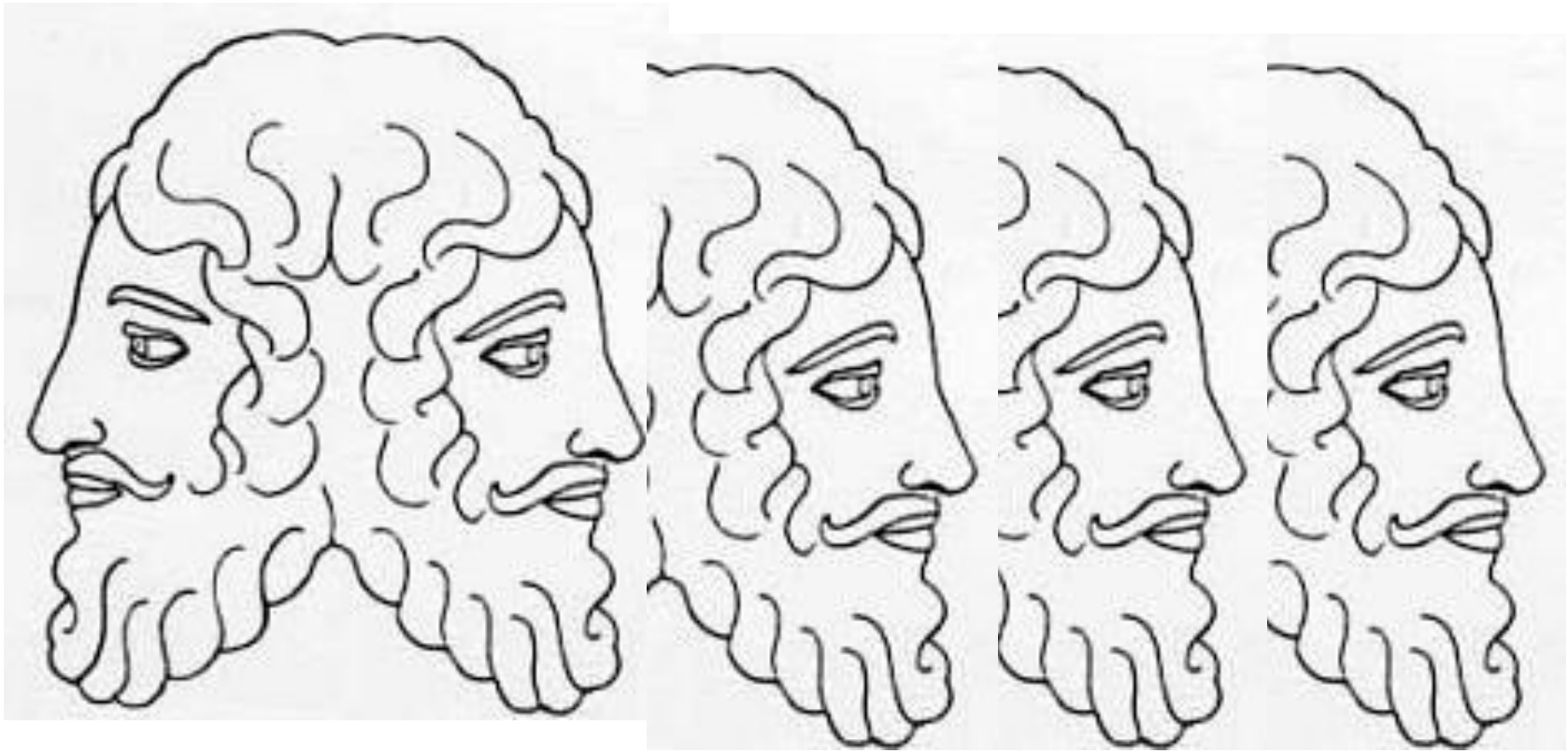
Reliability: *does* what is expected

Reliability: *measured in…*

success/failure **probability**

Mean **T**ime **T**o **F**ailure

Unreliability: *doesn't* do what is expected

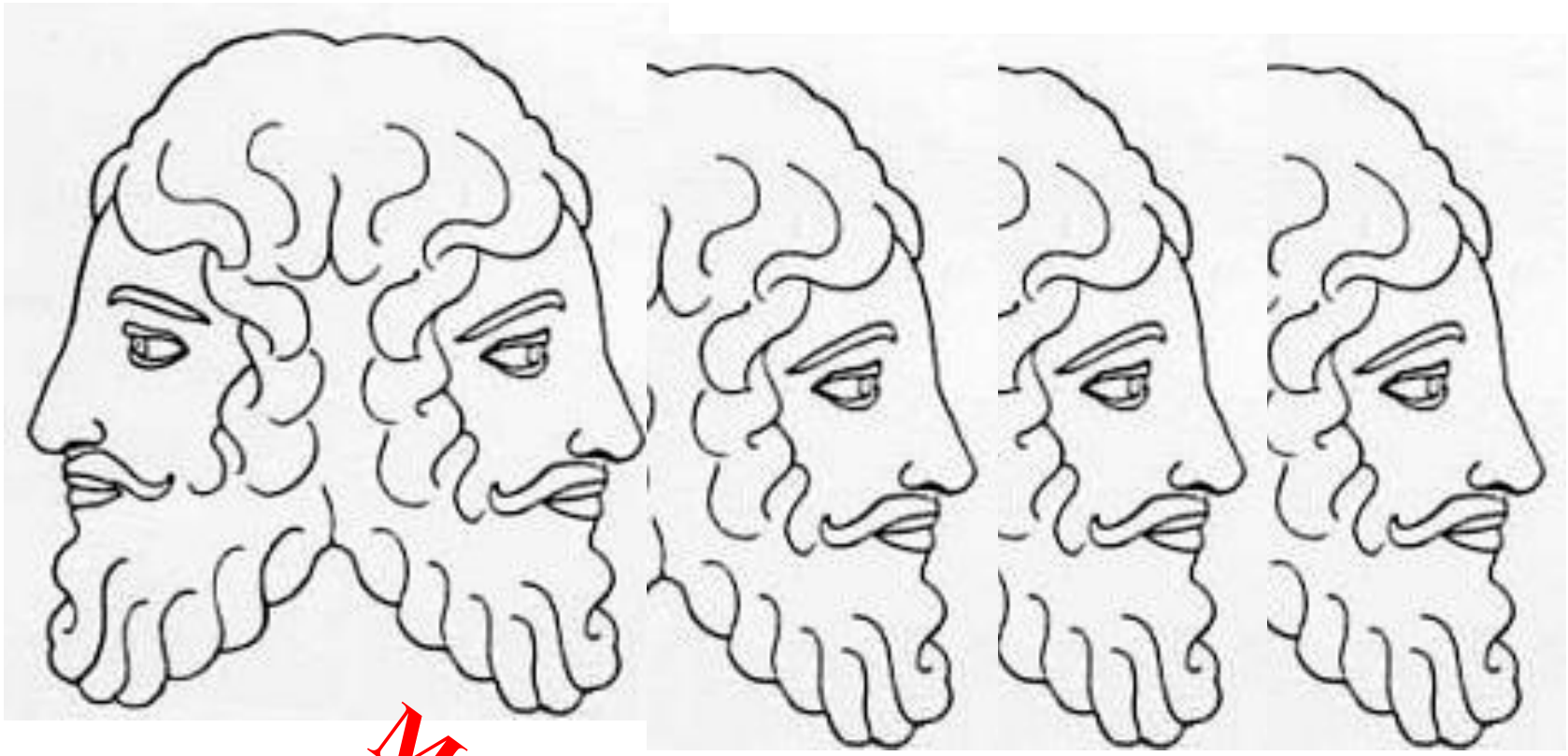


unavailable

incorrect

compromised

unsafe



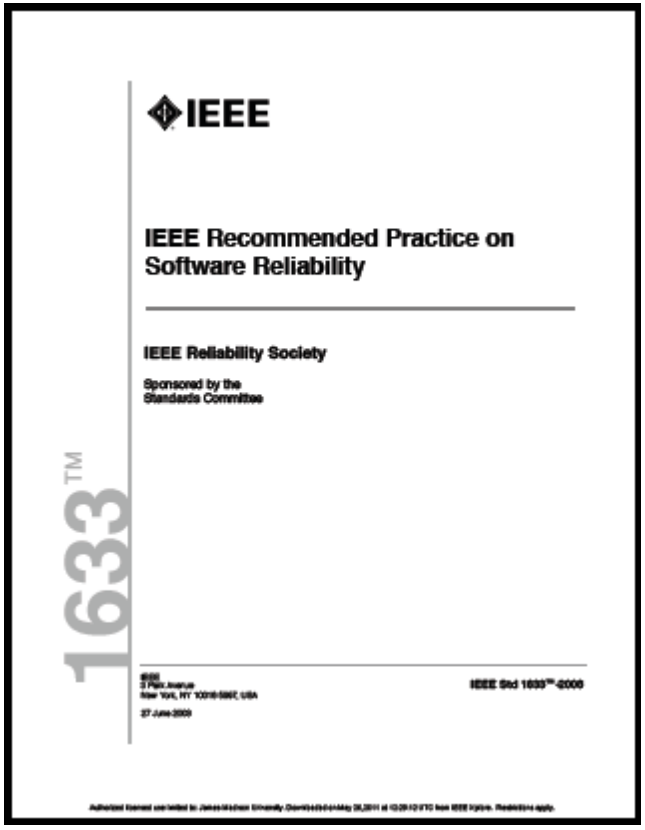
measured in …

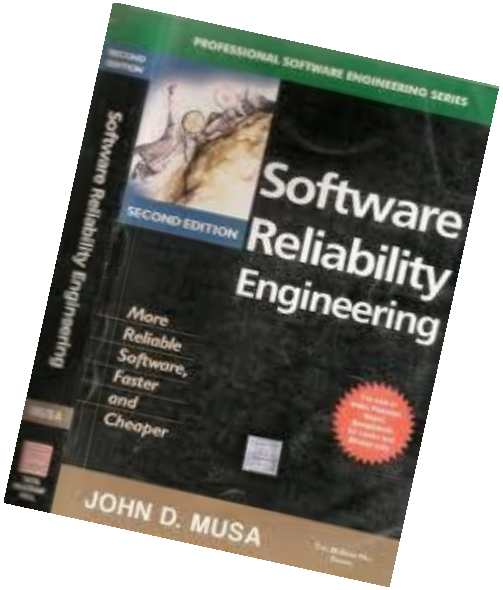
Mean Time To Repair

risk exposure

mission failure

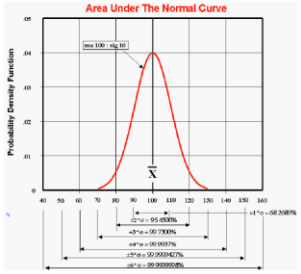
$ loss


IEEE
IEEE Recommended Practice on Software Reliability
IEEE Reliability Society
Sponsored by the Standards Committee
1633™
IEEE Std 1633™-2008
27 June 2008


PROFESSIONAL SOFTWARE ENGINEERING SERIES
Software Reliability Engineering
SECOND EDITION
Software Reliability Engineering
More Reliable Software, Faster and Cheaper
JOHN D. MUSA


HANDBOOK OF
SOFTWARE RELIABILITY ENGINEERING
MICHAEL R. LYU, EDITOR

Software Reliability Engineering

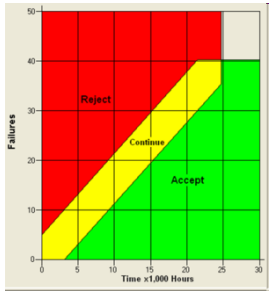


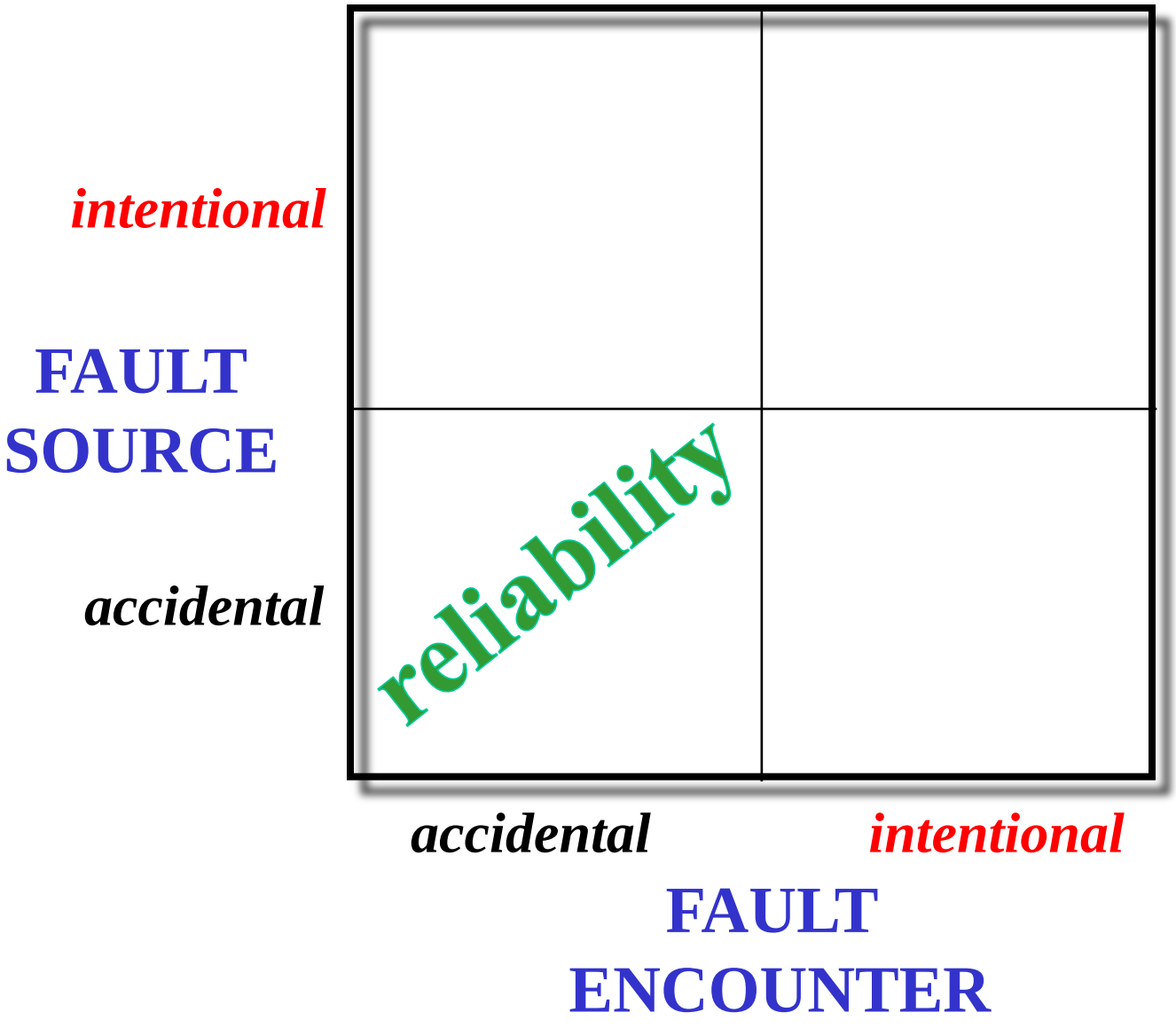
Establish quantitative reliability targets

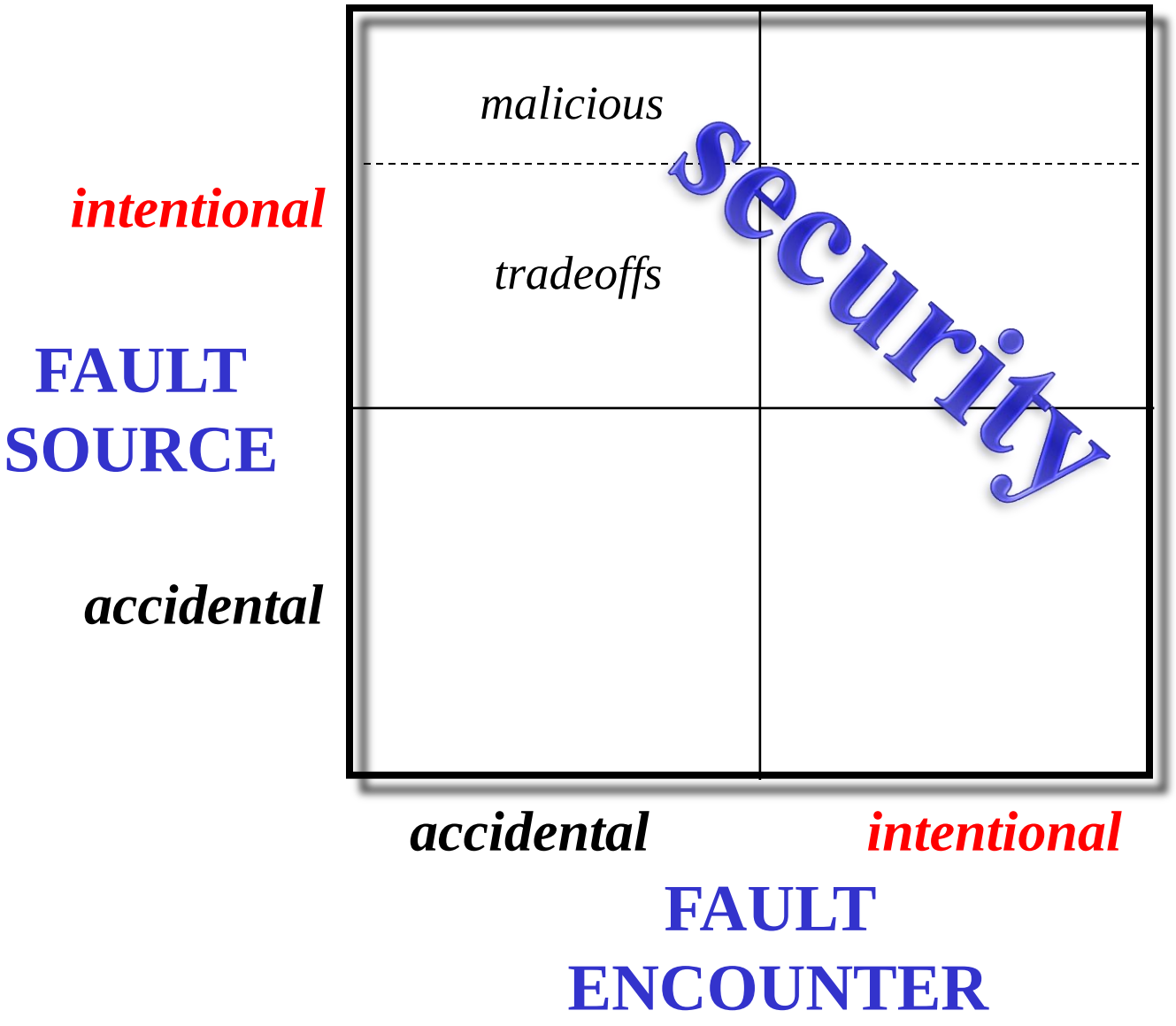
Construct usage profiles of operational system

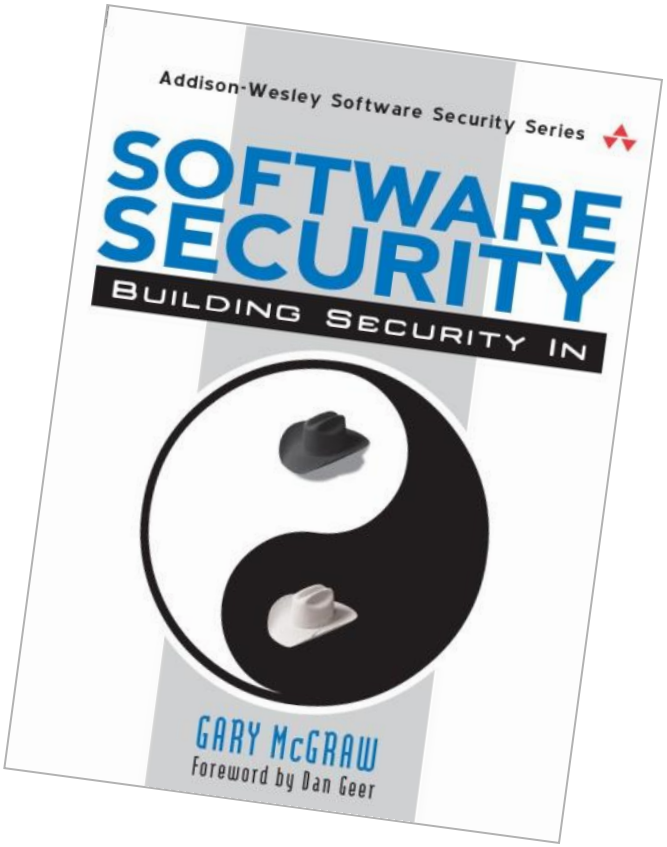
Operation	Occurrence probability	Initial test cases
Enter card	.332	66
Verify PIN	.332	66
Withdraw checking	.199	40
Withdraw savings	.066	13
Deposit checking	.040	8
Deposit savings	.020	4
Query status	.00664	1
Test terminal	.00332	1
Input to stolen card list	0.00058	0
Backup files	0.000023	0
Total	1	199

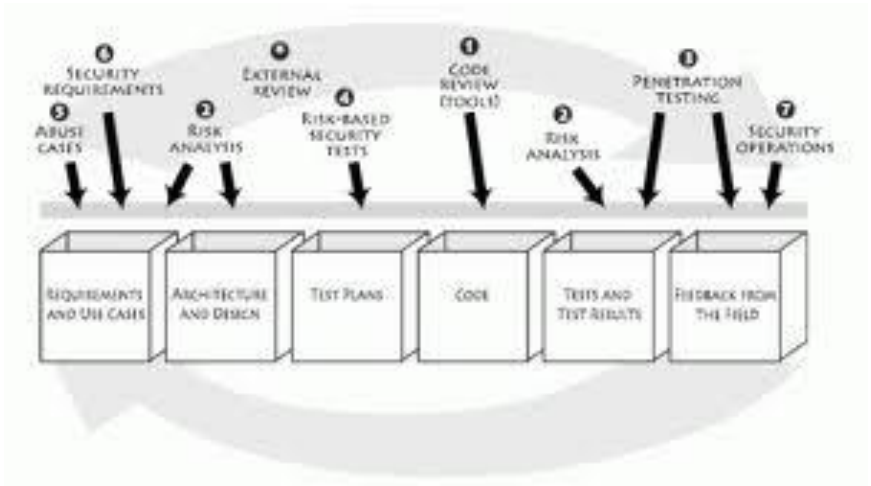
Test statistically to predict system reliability

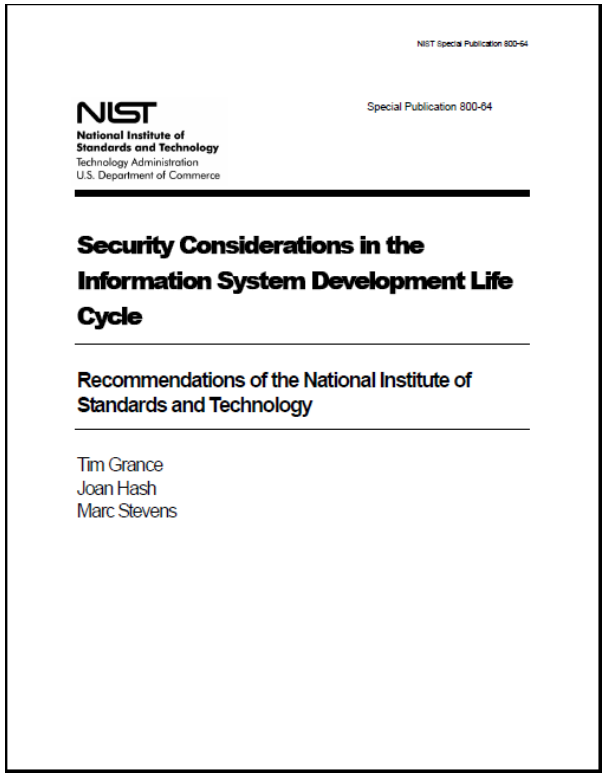



intentional
FAULT SOURCE
accidental
reliability
accidental
intentional
FAULT ENCOUNTER


malicious
tradeoffs
security
intentional
FAULT
SOURCE
accidental
accidental
intentional
FAULT
ENCOUNTER






NIST Special Publication 800-64

NIST
National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

Special Publication 800-64

Security Considerations in the Information System Development Life Cycle

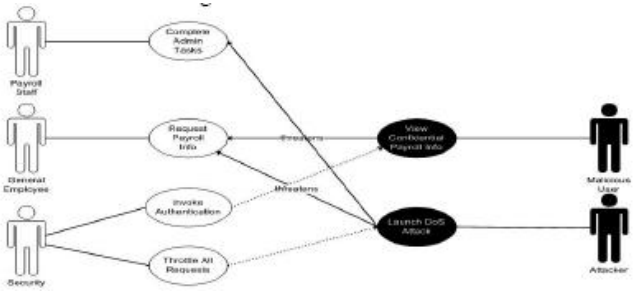
Recommendations of the National Institute of Standards and Technology

Tim Grance
Joan Hash
Marc Stevens

Software Security Engineering

Establish multiple quantitative targets

Use threat modeling to identify abuse cases



Rethink software reliability growth modeling

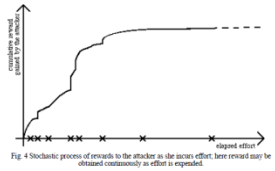
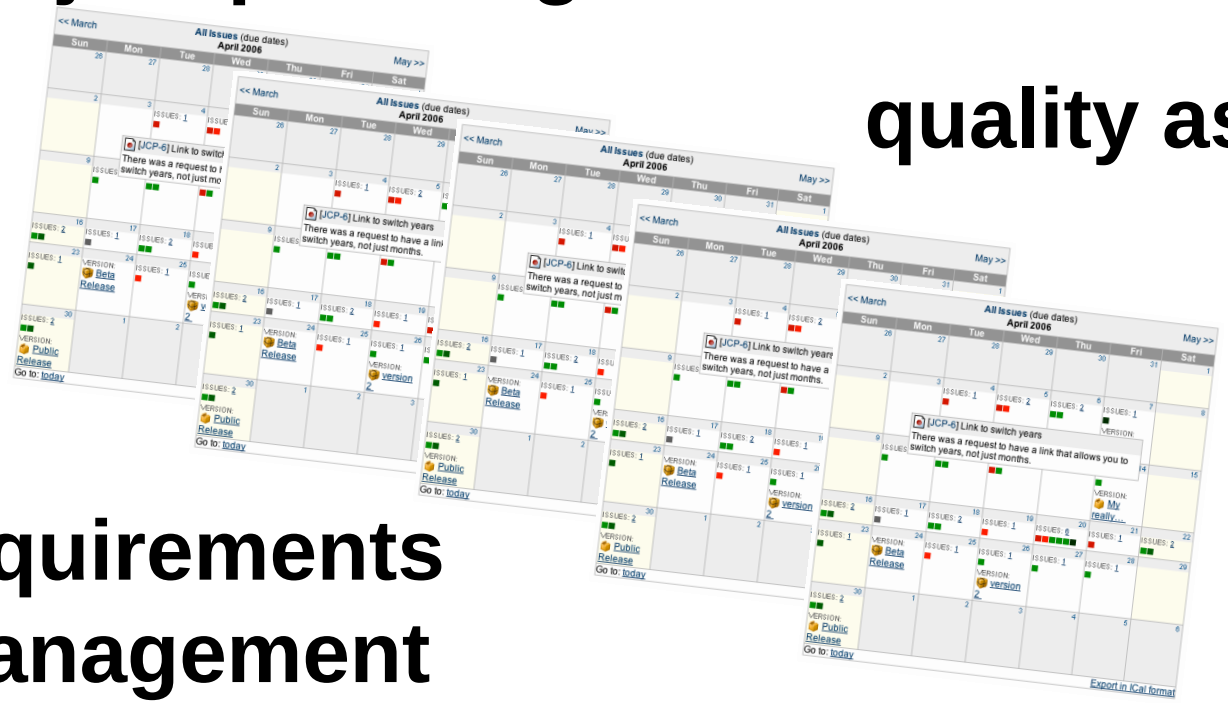


Fig. 4 Stochastic process of rewards to the attacker as she incurs effort; here reward may be obtained continuously as effort is expended

Attack	Occurrence probability	Initial test cases
Enter card	.332	66
Verify PIN	.332	66
Withdraw checking	.199	40
Withdraw savings	.066	13
Deposit checking	.040	8
Deposit savings	.020	4
Query status	.00664	1
Test terminal	.00332	1
Input to stolen card list	0.00058	0
Backup files	0.000023	0
Total	1	199

project planning

quality assurance



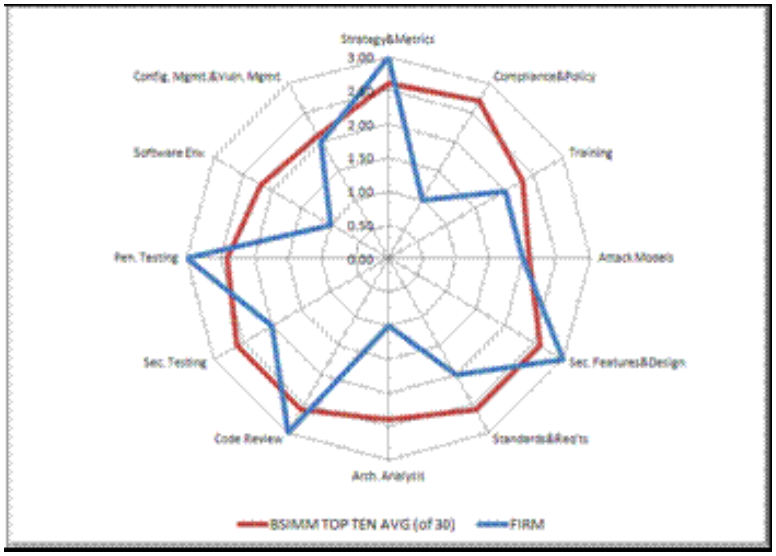
project tracking

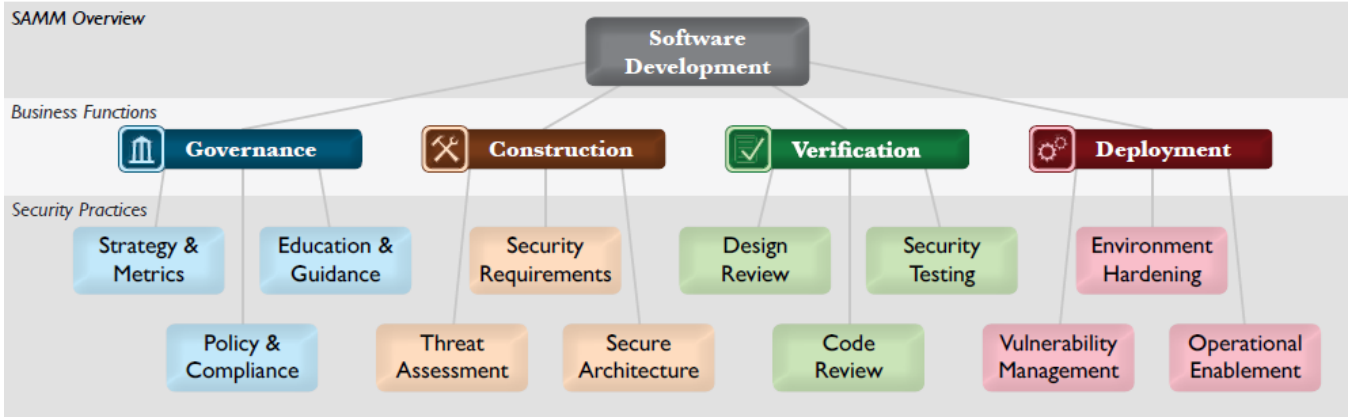
requirements management

configuration management

Key Process Areas

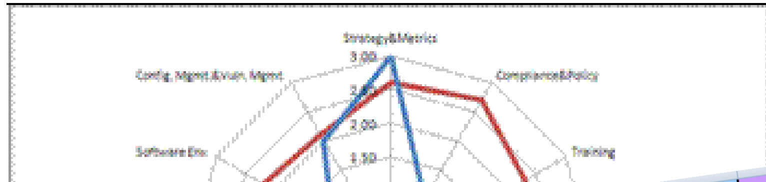
Build
Security
In
Maturity
Model

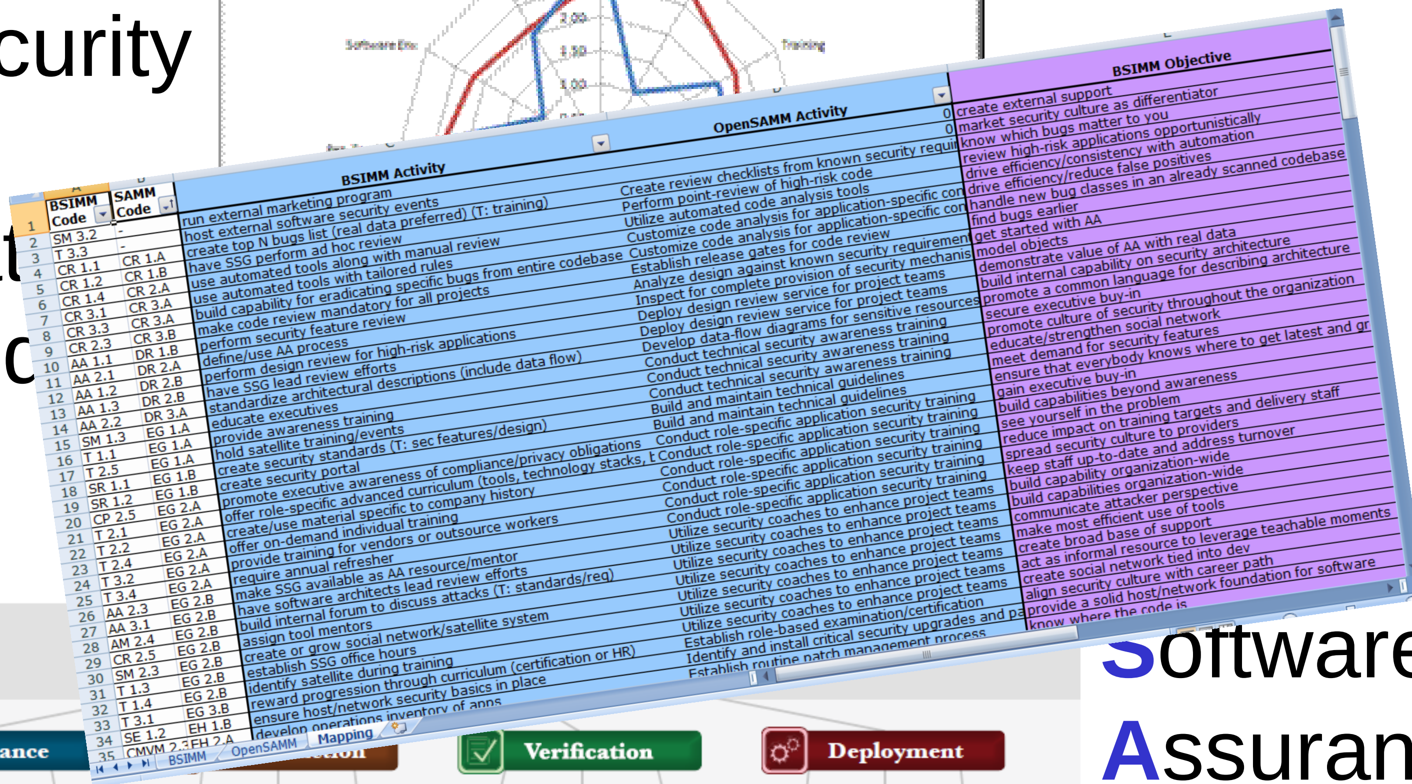




Software
Assurance
Maturity
Model

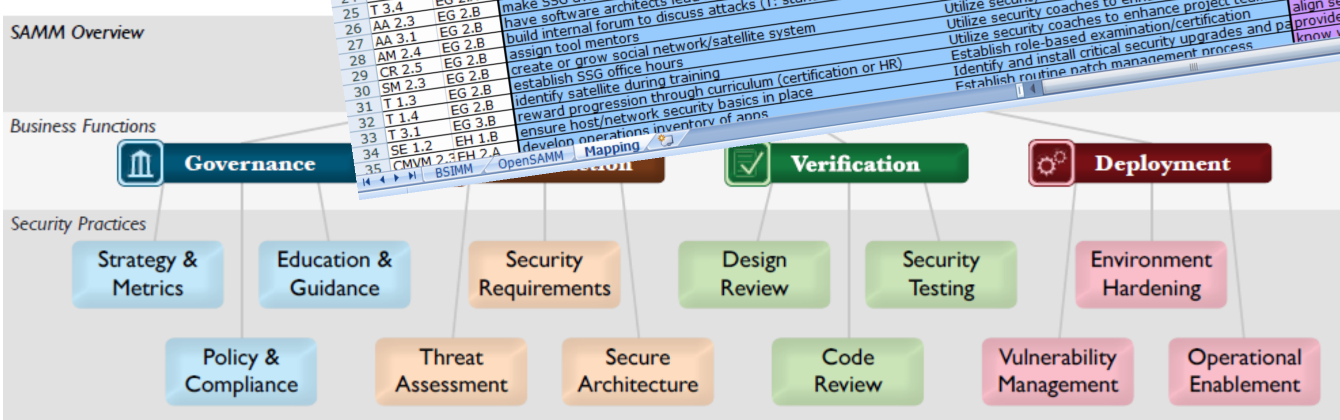
Build
Security
In
Mat
Moc





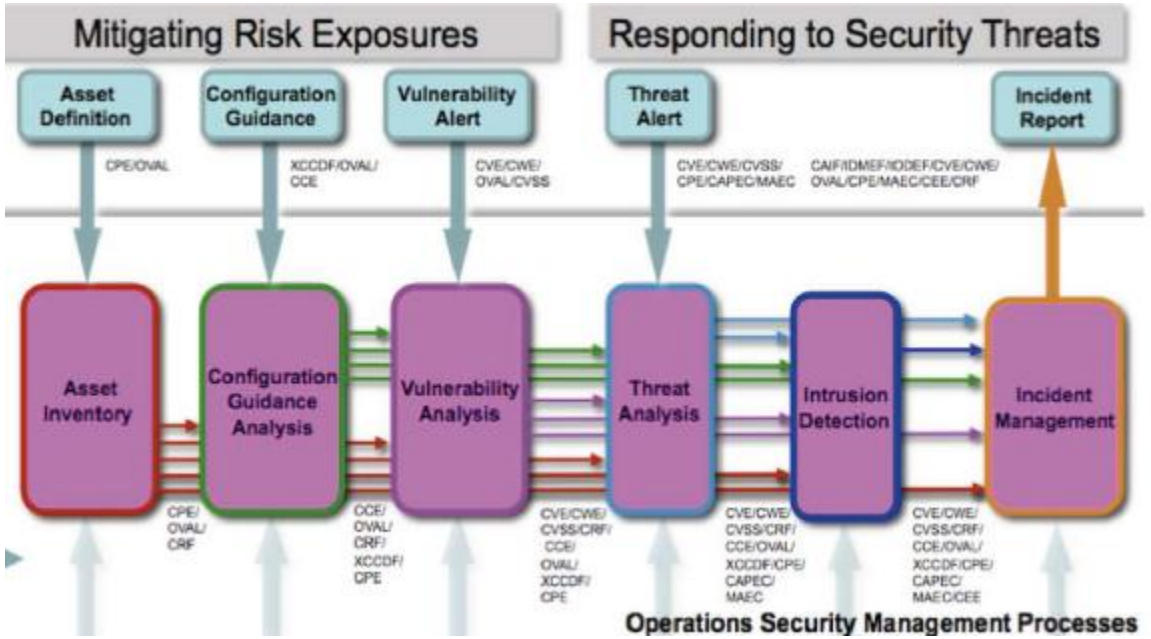
	BSIMM Code	SAMM Code	BSIMM Activity	OpenSAMM Activity	BSIMM Objective
2	SM 3.2	-	run external marketing program	0	create external support
3	T 3.3	-	host external software security events	0	market security culture as differentiator
4	CR 1.1	CR 1.A	create top N bugs list (real data preferred) (T: training)	Create review checklists from known security requi	know which bugs matter to you
5	CR 1.2	CR 1.B	have SSG perform ad hoc review	Perform point-review of high-risk code	review high-risk applications opportunistically
6	CR 1.4	CR 2.A	use automated tools along with manual review	Utilize automated code analysis tools	drive efficiency/consistency with automation
7	CR 3.1	CR 3.A	use automated tools with tailored rules	Customize code analysis for application-specific con	drive efficiency/reduce false positives
8	CR 3.3	CR 3.A	build capability for eradicating specific bugs from entire codebase	Customize code analysis for application-specific con	handle new bug classes in an already scanned codebase
9	CR 2.3	CR 3.B	make code review mandatory for all projects	Establish release gates for code review	find bugs earlier
10	AA 1.1	DR 1.B	perform security feature review	Analyze design against known security requirement	get started with AA
11	AA 2.1	DR 2.A	define/use AA process	Inspect for complete provision of security mechanis	model objects
12	AA 1.2	DR 2.B	perform design review for high-risk applications	Deploy design review service for project teams	demonstrate value of AA with real data
13	AA 1.3	DR 2.B	have SSG lead review efforts	Deploy design review service for project teams	build internal capability on security architecture
14	AA 2.2	DR 3.A	standardize architectural descriptions (include data flow)	Develop data-flow diagrams for sensitive resources	promote a common language for describing architecture
15	SM 1.3	EG 1.A	educate executives	Conduct technical security awareness training	secure executive buy-in
16	T 1.1	EG 1.A	provide awareness training	Conduct technical security awareness training	promote culture of security throughout the organization
17	T 2.5	EG 1.A	hold satellite training/events	Conduct technical security awareness training	educate/strengthen social network
18	SR 1.1	EG 1.B	create security standards (T: sec features/design)	Build and maintain technical guidelines	meet demand for security features
19	SR 1.2	EG 1.B	create security portal	Build and maintain technical guidelines	ensure that everybody knows where to get latest and gr
20	CP 2.5	EG 2.A	promote executive awareness of compliance/privacy obligations	Conduct role-specific application security training	gain executive buy-in
21	T 2.1	EG 2.A	offer role-specific advanced curriculum (tools, technology stacks, b	Conduct role-specific application security training	build capabilities beyond awareness
22	T 2.2	EG 2.A	create/use material specific to company history	Conduct role-specific application security training	see yourself in the problem
23	T 2.4	EG 2.A	offer on-demand individual training	Conduct role-specific application security training	reduce impact on training targets and delivery staff
24	T 3.2	EG 2.A	provide training for vendors or outsource workers	Conduct role-specific application security training	spread security culture to providers
25	T 3.4	EG 2.A	require annual refresher	Conduct role-specific application security training	keep staff up-to-date and address turnover
26	AA 2.3	EG 2.B	make SSG available as AA resource/mentor	Utilize security coaches to enhance project teams	build capability organization-wide
27	AA 3.1	EG 2.B	have software architects lead review efforts	Utilize security coaches to enhance project teams	build capabilities organization-wide
28	AM 2.4	EG 2.B	build internal forum to discuss attacks (T: standards/req)	Utilize security coaches to enhance project teams	communicate attacker perspective
29	CR 2.5	EG 2.B	assign tool mentors	Utilize security coaches to enhance project teams	make most efficient use of tools
30	SM 2.3	EG 2.B	create or grow social network/satellite system	Utilize security coaches to enhance project teams	create broad base of support
31	T 1.3	EG 2.B	establish SSG office hours	Utilize security coaches to enhance project teams	act as informal resource to leverage teachable moments
32	T 1.4	EG 2.B	identify satellite during training	Utilize security coaches to enhance project teams	create social network tied into dev
33	T 3.1	EG 3.B	reward progression through curriculum (certification or HR)	Establish role-based examination/certification	align security culture with career path
34	SE 1.2	EH 1.B	ensure host/network security basics in place	Identify and install critical security upgrades and pa	provide a solid host/network foundation for software
35	CMVM 2.3	EH 2.A	develop operations inventory of apps	Establish routine patch management process	know where the code is

BSIMM | OpenSAMM | Mapping



Software
Assurance
Maturity
Model

The Software Security Framework (SSF)			
Governance	**Intelligence**	**SSDL Touchpoints**	**Deployment**
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

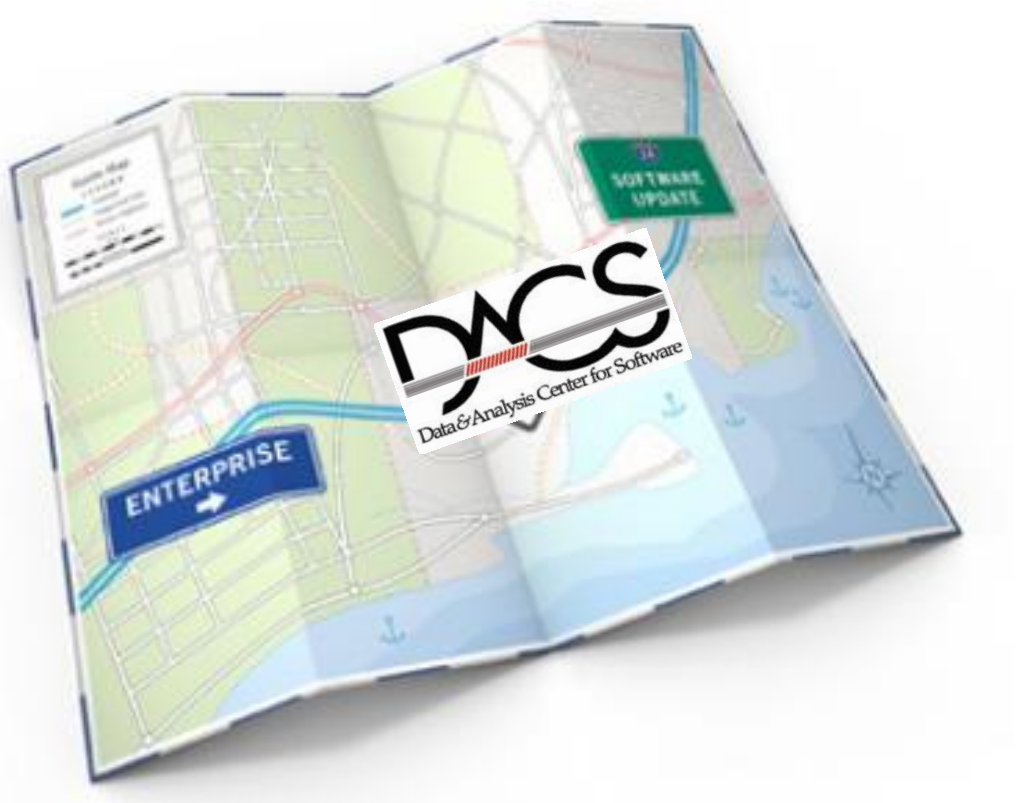


PROTECTING SENSITIVE COMPARTMENTED INFORMATION WITHIN INFORMATION SYSTEMS (DCID 6/3)—MANUAL

DACS Software Reliability Initiative

= *"Roadmap to Dependability"*

START

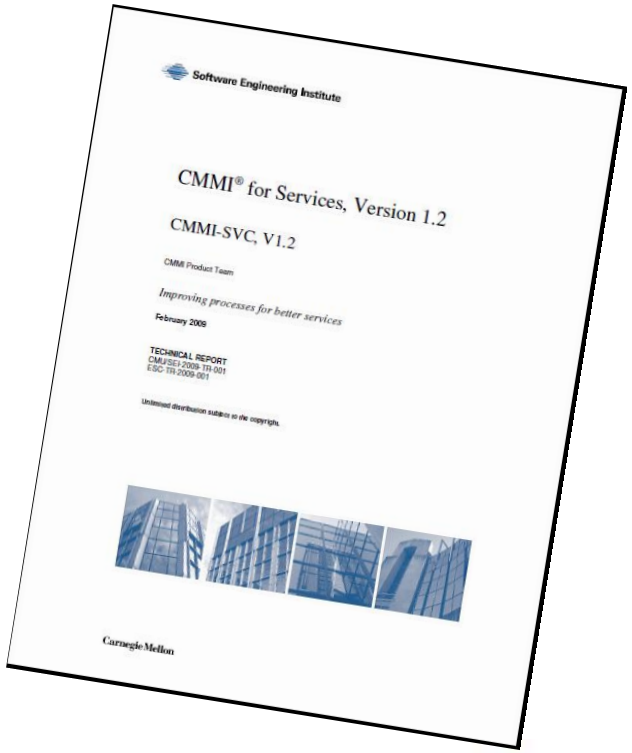

SOFTWARE UPDATE
ENTERPRISE
DACS
Data&Analysis Center for Software

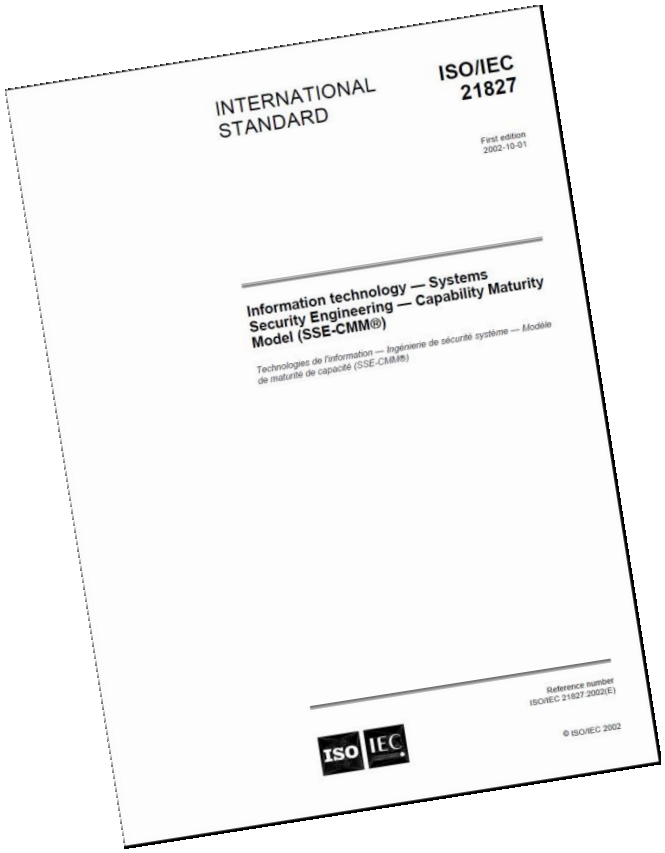

You are Here


EAST BOULEVARD
EUCLID
CORNELL
ADELBERT
Lagoon
Freiberger Field
Mather Dance
Mather House
Church of the Covenant
Kelvin Smith Library
Thwing
Cleveland Hearing & Speech
Adelbert
Pathology
You are here.

Evaluation of existing capabilities

CMMI


Software Engineering Institute
CMMI® for Services, Version 1.2
CMMI-SVC, V1.2
CMMI Product Team
Improving processes for better services
February 2009
TECHNICAL REPORT
CMU/SEI-2009-TR-001
ESC-TR-2009-001
Unlimited distribution subject to the copyright.
Carnegie Mellon


INTERNATIONAL STANDARD
ISO/IEC 21827
First edition
2002-10-01
Information technology — Systems Security Engineering — Capability Maturity Model (SSE-CMM®)
Technologies de l'information — Ingénierie de sécurité système — Modèle de maturité de capacité (SSE-CMM®)
Reference number
ISO/IEC 21827:2002(E)
© ISO/IEC 2002

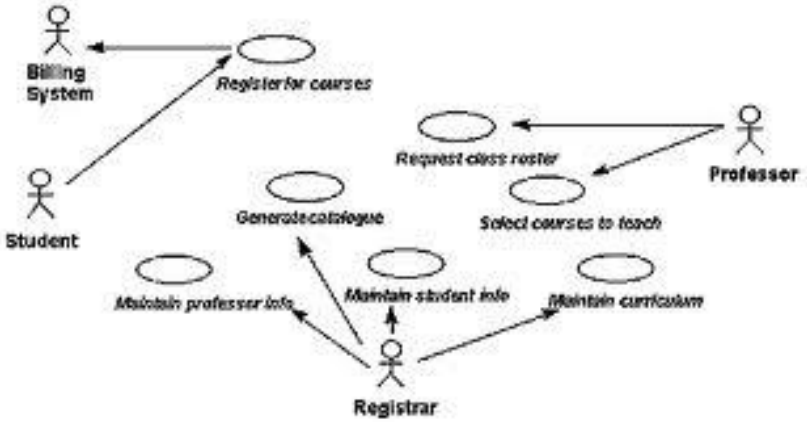
ISO IEC


OPTIMIZING
5
PREDICTABLE
4
ESTABLISHED
3
MANAGED
2
PERFORMED
1
INCOMPLETE
0

END

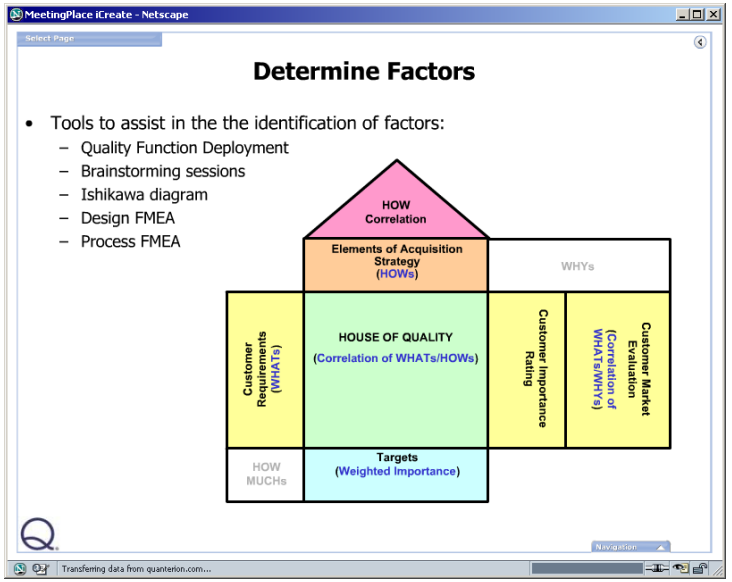

SOFTWARE
UPDATE
ENTERPRISE
DACS
Data&Analysis Center for Software

Specification of requirements


Billing System
Register for courses
Request class roster
Professor
Student
Generate catalogue
Select courses to teach
Maintain professor info
Maintain student info
Maintain curriculum
Registrar


Acceptance Test Plan
Template


SOFTWARE REQUIREMENTS ENGINEERING
Second Edition
IEEE


MeetingPlace iCreate - Netscape
Determine Factors
• Tools to assist in the the identification of factors:
– Quality Function Deployment
– Brainstorming sessions
– Ishikawa diagram
– Design FMEA
– Process FMEA
HOW Correlation
Elements of Acquisition Strategy (HOWs)
WHYs
Customer Requirements (WHATs)
HOUSE OF QUALITY
(Correlation of WHATs/HOWs)
Customer Importance Rating
Customer Market Evaluation
(Correlation of WHATs/WHYs)
HOW MUCHs
Targets
(Weighted Importance)
Transferring data from quanterion.com...

How to get from *here* to *there*

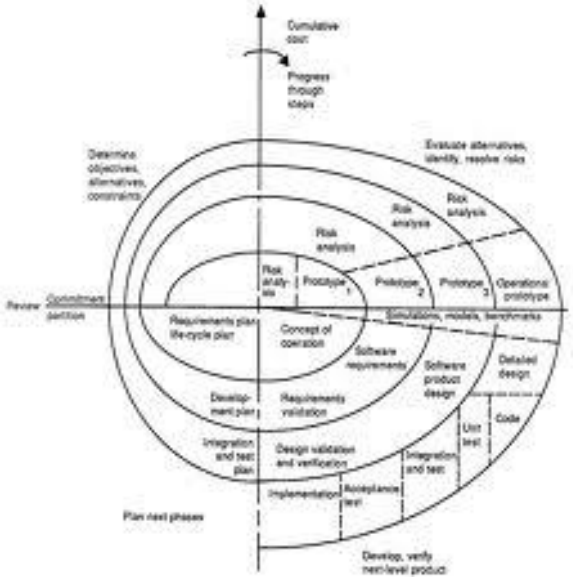
Recommended + deprecated practices

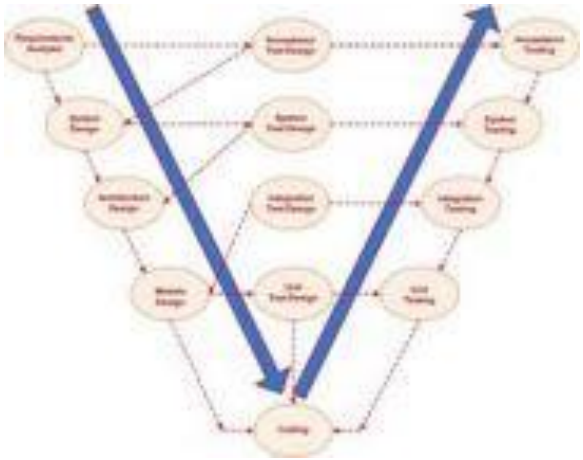
Guide to the Software Engineering Body of Knowledge

2004 Version

SWEBOK®

A project of the IEEE Computer Society Professional Practices Committee





NIST
National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

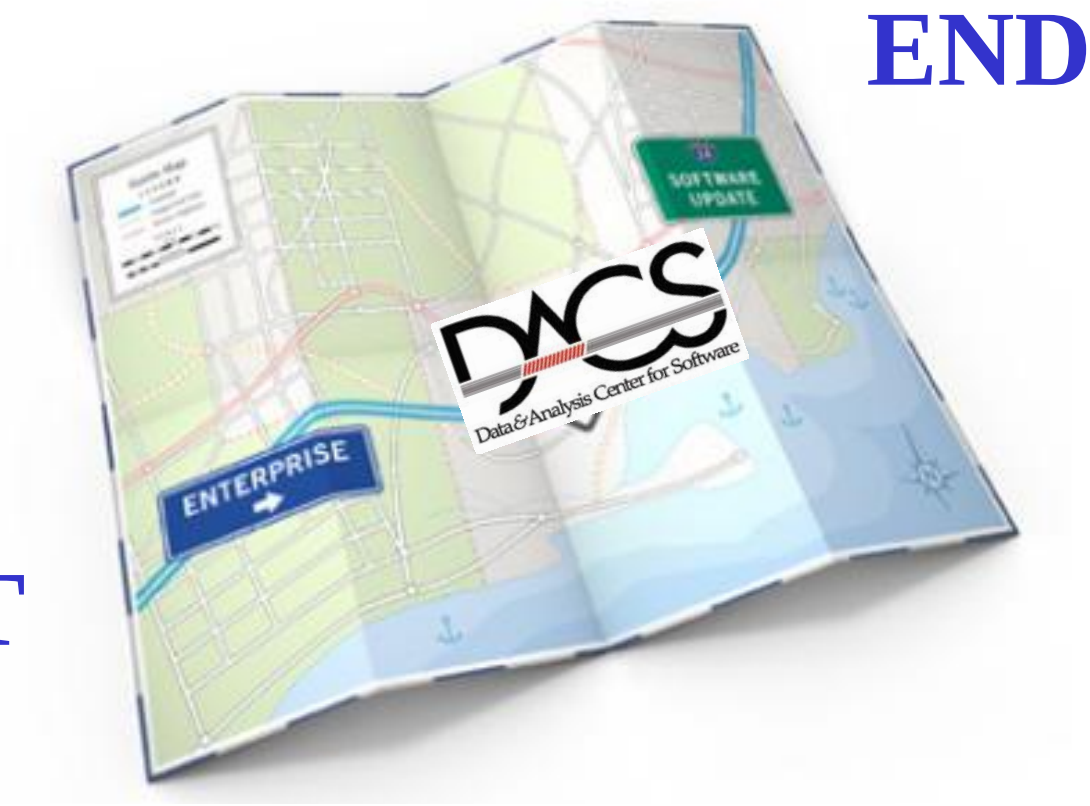
Special Publication 800-64

Security Considerations in the Information System Development Life Cycle

Recommendations of the National Institute of Standards and Technology

Tim Grance
Joan Hash
Marc Stevens

END



START

How to get from ***here*** to ***there***

END

START

How to get from ***here*** to ***there***


“Are we there yet?”
ENTERPRISE
SOFTWARE UPDATE

Assessment Techniques

NASA-STD-8739.8

NOT MEASUREMENT SENSITIVE

National Aeronautics and Space Administration

NASA-STD-8739.8 w/Change 1
July 28, 2004

SOFTWARE ASSURANCE STANDARD

NASA TECHNICAL STANDARD

REPLACES NASA-STD-2201-93 DATED NOVEMBER 1992

IEEE Standards

IEEE Std 1012™-2004
(Revision of
IEEE Std 1012-1998)

1012™

IEEE Standard for Software Verification and Validation

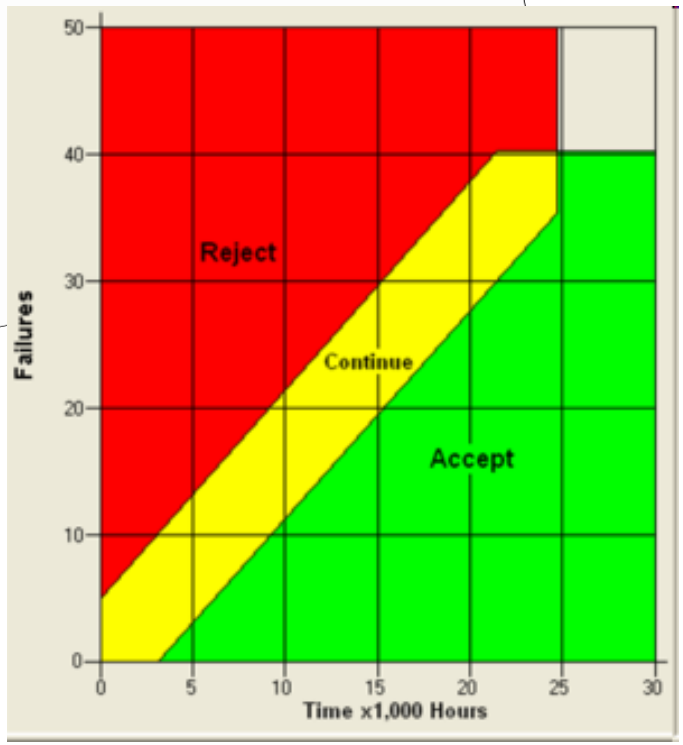
IEEE Computer Society

Sponsored by the
Software Engineering Standards Committee

IEEE

3 Park Avenue, New York, NY10016-5997, USA

8 June 2005
Print: SH95308
PDF: SS95308



DACS Software Reliability Initiative

= *"Roadmap to Dependability"*

Securing Systems through Software Reliability Engineering

Taz Daughtrey

hdaughtrey@theDACS.com

www.thedacs.com